

Article

Virtual Sensing and Sensors Selection for Efficient Temperature Monitoring in Indoor Environments [†]

Andrea Brunello ^{1,*}, Andrea Urgolo ^{1,*}, Federico Pittino ², András Montvay ² and Angelo Montanari ¹

¹ Data Science and Automatic Verification Laboratory, University of Udine, Via delle Scienze 206, 33100 Udine, Italy; angelo.montanari@uniud.it

² Silicon Austria Labs GmbH, Europastraße 12, A-9524 Villach, Austria; federico.pittino@silicon-austria.com (F.P.); andras.montvay@silicon-austria.com (A.M.)

* Correspondence: andrea.brunello@uniud.it (A.B.); andrea.urgolo@uniud.it (A.U.)

† This paper is an extension of a conference work: Brunello, A.; Kraft, M.; Montanari, A.; Pittino, F.; Urgolo, A. Virtual Sensing of Temperatures in Indoor Environments: A Case Study. In Proceedings of the 2020 International Conference on Data Mining Workshops (ICDMW) (pp. 805–810). Sorrento, Italy, 17–20 November 2020.

‡ These authors contributed equally to this work.

Abstract: Real-time estimation of temperatures in indoor environments is critical for several reasons, including the upkeep of comfort levels, the fulfillment of legal requirements, and energy efficiency. Unfortunately, setting an adequate number of sensors at the desired locations to ensure a uniform monitoring of the temperature in a given premise may be troublesome. Virtual sensing is a set of techniques to replace a subset of physical sensors by virtual ones, allowing the monitoring of unreachable locations, reducing the sensors deployment costs, and providing a fallback solution for sensor failures. In this paper, we deal with temperature monitoring in an open space office, where a set of physical sensors is deployed at uneven locations. Our main goal is to develop a black-box virtual sensing framework, completely independent of the physical characteristics of the considered scenario, that, in principle, can be adapted to any indoor environment. We first perform a systematic analysis of various distance metrics that can be used to determine the best sensors on which to base temperature monitoring. Then, following a genetic programming approach, we design a novel metric that combines and summarizes information brought by the considered distance metrics, outperforming their effectiveness. Thereafter, we propose a general and automatic approach to the problem of determining the best subset of sensors that are worth keeping in a given room. Leveraging the selected sensors, we then conduct a comprehensive assessment of different strategies for the prediction of temperatures observed by physical sensors based on other sensors' data, also evaluating the reliability of the generated outputs. The results show that, at least in the given scenario, the proposed black-box approach is capable of automatically selecting a subset of sensors and of deriving a virtual sensing model for an accurate and efficient monitoring of the environment.

Keywords: virtual sensing; sensor selection; temperature monitoring; machine learning; neural networks; particle filters; distance metrics



Citation: Brunello, A.; Urgolo, A.; Pittino, F.; Montvay, A.; Montanari, A. Virtual Sensing and Sensors Selection for Efficient Temperature Monitoring in Indoor Environments. *Sensors* **2021**, *21*, 2728. <https://doi.org/10.3390/s21082728>

Academic Editor: Tomoyuki Yokota

Received: 8 March 2021

Accepted: 6 April 2021

Published: 13 April 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Sensing means to grasp the nature, significance, or meaning of a physical phenomenon or stimulus. A sensor can be defined as a device which measures a physical quantity and transforms it into a signal that can be interpreted by an instrument or an observer. Sensors can be partitioned into two main classes: physical sensors, which measure physical phenomena directly, and virtual sensors, which are designed to process various physically measured data and to produce values approximating those that would be directly reported by physical sensors [1].

Virtual sensing relies on data captured by physical sensors and on the use of suitable models to combine them in order to estimate a difficult or expensive-to-measure quantity. Limitations may indeed be present regarding where actual sensors may be placed, and they can be either of physical or of economical nature. For this reason, over the past decades, virtual sensors have been successfully developed and adopted in various fields, exploiting any indirect approach that relies on the outputs of other physical or virtual sensors, process models, and property relations [2].

One of the many applications of virtual sensing is connected to the accurate measurement of temperature in an indoor environment. Such a scenario comprises a large variety of applications, ranging from control of the comfort and energy efficiency in an office or residential space to the monitoring of safe temperature levels in an industrial environment. To obtain an accurate temperature measurement, a very large number of sensors typically need to be deployed in the environment, covering all possible heat sources and sinks, and monitoring heat propagation. Virtual sensing is a promising technique for decreasing the complexity of the network of monitoring physical sensors by replacing some of them with virtual ones, in which the temperature value is inferred from the physical measurements, while retaining measurement accuracy and fine spatio-temporal granularity. This technique, however, poses several challenges, namely how to choose the location of the physical sensors, how many of them are needed, and finding the best prediction method for optimal accuracy. In this paper, we address these issues, taking the temperature monitoring in an open space office at Silicon Austria Labs premise in Villach (Austria) as a case study.

The approach we developed includes, as a first step, the selection of a subset of sensors on which the prediction is based. Such a selection is made by evaluating several distance metrics, including a novel metric obtained by a combination of existing ones through a genetic programming approach. The proposed solution is actually a black-box one—that is, it is designed with the idea of making it easily generalizable to any other indoor environment. To achieve this, the presence of elements in the room capable of influencing the internal temperature, like windows and radiators, was not modeled directly, but through the automatic selection of sensors adjacent to them, which should carry useful information for the task of temperature prediction. Afterwards, based on the developed metric and on a set of thoroughly engineered features, we performed a comprehensive evaluation of several temperature prediction techniques, which include particle filters, machine learning (in particular, linear regression and ensembles), and deep learning solutions. In order to obtain an indication of the reliability of the generated predictions, we also carried out a study based on the estimation of confidence intervals. Overall, the devised approach can be considered as an automatic black-box methodology to select the number and position of the sensors to be retained and the ones that will be virtualized.

The paper is organized as follows. Section 2 gives a short account of related work on virtual sensing and sensor selection. Then, Section 3 introduces the considered case study. Next, we turn to the experimental part of the work, which consists of three sections. We begin with Section 4, where we perform a preliminary descriptive analysis of our dataset. Then, in Section 5, the data pre-processing phase is described, along with the feature engineering step, the development of a novel sensor distance metric based on a genetic programming approach, and the proposal of a domain-independent sensor selection technique. On the basis of the results of the pre-processing tasks, Section 6 thoroughly describes the temperature prediction experiments, and illustrates the main results, including a study of the prediction's reliability. In addition, a comparison with respect to the performance achievable by means of a brute force sensor selection approach is carried out. Section 7 concludes the main body of the paper with a summary of the main take-home messages and an outline of a high-level procedure to replicate the proposed approach in any indoor environment. Finally, Section 8 provides an assessment of the work done and outlines directions for future work.

2. Related Work

The present work studies the problems of virtual sensing and sensor subset selection. Accordingly, we start the analysis of related work with a brief introduction to virtual sensing by analyzing some of the most important approaches proposed in the literature, then we focus on sensor subset selection considering the various application areas where the problem has been addressed.

Over the years, virtual sensing has found application in several domains including robotics, automation, anomaly/leak detection, air quality control, active noise suppression, wireless communication, automotive, and transportation [1–3]. In the literature, three main virtual sensing modeling methods have been taken into account: (i) *first-principle*, where virtual sensors are developed by hand on the basis of fundamental laws of physics and an extensive domain knowledge; (ii) *black-box*, where models capable of exploiting empirical correlations present in the data are built, without any knowledge of the underlying physical processes (most of the statistical and machine learning methods belong to this category); (iii) *grey-box*, which exploits a combination of first-principle and black-box approaches. In this work, we focus on the latter two approaches, in order to develop generic models that do not require complex domain knowledge and are not tied to the specific use case.

As for black-box techniques, the estimation of a variable at locations where it has not been observed by exploiting the data available at other locations can be made by means of approaches commonly used for the interpolation of scatter points, such as Inverse Distance Weighting (IDW) and Kriging [4]. IDW assumes that the interpolating surface is more influenced by closer points than distant ones, while Kriging is a geostatistical regression method used in spatial analysis that spatially interpolates a quantity minimizing the mean squared error. As it comes out, both approaches find difficult application in real-world settings, typically being affected by spatial-temporal anisotropy and unable to deal with non-linear relations among predictor variables. When partial observations are made and random perturbations are present in the data, methods such as Kalman filters and particle filters [5] can be relied on for the estimation of the internal states of dynamical systems. These methods compute the unknown quantities through posterior distributions obtained from Bayesian inference models [6,7]. Although particle filters are not subject to the constraints of non-Gaussianity of perturbations and linearity of the dynamic systems that affect Kalman ones, they have some disadvantages as well, related to possible resampling biases and to the coarseness of the definition of the likelihood distribution, which may be unable to capture all relevant real-world characteristics. To overcome the limitations of these techniques, some authors explored approaches based on machine learning, like support vector machines, decision trees, and ensembles (such as random forests) [8–10]. More recently, deep learning solutions have been considered as well [11–14], and were deemed able to exploit implicit information on temporal trends and spatial associations among sensors. As an example, in [15,16] the authors employ a Long Short-Term Memory (LSTM) network and a combined Convolutional LSTM (ConvLSTM) network, respectively, capable of learning from long-term dependencies in spatial-temporal information. Despite the latter ConvLSTM-based model achieving a good performance, it is not applicable in context like ours, where the positions of sensors are fixed and quite sparse.

As for the sensor subset selection problem, different approaches have been explored in the literature in various domains. In [17], an optimal sensor selection and fusion solution is proposed for the case of an heat exchanger fouling diagnosis in aerospace systems. It is based on the minimum Redundancy Maximum Relevance (mRMR) algorithm, and it can be applied only for classification tasks with discretized features. Another sensor selection approach for optimal Fault Detection and Isolation (FDI) tests in complex systems is presented in [18]. In this case, steady-state or dynamic models are assumed to be available, and the estimation is based on their contribution to information gain through Hellinger distance (HD) and Kullback–Leibler divergence (KLD).

In the context of mobile crowd sensing (MCS) systems, which leverage a public crowd equipped with various mobile devices for large-scale sensing tasks, a solution based

on reverse combinatorial auctions is proposed in [19], which integrates the concepts of social welfare, quality of information, and cost required by each single user to provide an observation, in order to select an optimal subset of users from whom it is convenient to request data in exchange for a reward. This approach is not applicable to our case, as both the quality level of the sensors and the cost of each observation are the same.

A different class of problems where sensor selection techniques can be exploited is that of sensor scheduling problems, where one or more sensors have to be selected at every time step. As an example, in the domain of linear dynamical systems, a greedy algorithm for sensor scheduling based on submodular error functions, is described in [20]. As another example, in the context of active robotic mapping, a technique to prune the search tree of all possible sensor schedules, based on a weighted function of the error covariances related to the state estimates, is illustrated in [21]. Finally, an algorithm for stochastic sensor selection minimizing the expected error covariance, based on Kalman filters with an underlying Hidden Markov Model, is proposed in [22]; it relies on the assumptions of process model linearity and sensor noise Gaussianity.

Finally, another domain where sensor selection techniques are of interest is that of Wireless Sensor Networks (WSN). WSN are networks extended over large geographical regions that feature low-power sensors which are interconnected to each other to receive and transmit data. In [23], an event-based sensor data scheduler deriving an approximate minimum mean-squared error (MMSE) estimator has been developed for remote state estimation over a network. In [24], the sensor selection problem over WSN was addressed using Kalman filters with various cost functions and network constraints and assuming a predetermined time horizon. Finally, a distributed sensor node-level energy management approach for minimizing energy usage was outlined in [25], which is based on target trajectory prediction through Kalman filters and Interacting Multiple Model (IMM) filters.

In contrast to all the above solutions, we propose a *black-box* approach to the selection of a subset of sensors, applicable to a generic indoor environment, and also well-suited for non-linear process models and time series consisting of both real and discrete values. Although we focus on the temperature estimation scenario, the approach can, in principle, be applied to any other prediction task.

3. The Considered Setting

The considered scenario is an open-space office at Silicon Austria Labs, in Villach (Austria). As shown in Figure 1, the room is fairly large, having an overall surface of 127 m², and it is characterized by the presence of an always-on air-conditioning system, intermittently used workplaces equipped with high-performance workstations, individually controlled radiators, and some windows, at the top and the bottom of the map, which can be independently opened and blinded. Twelve Raspberry Pi Zero boards (Figure 2) are deployed for the measurements, each one equipped with an Enviro pHAT sensor board featuring temperature, pressure, light, color, motion, and analog sensors. The recordings are transmitted via WLAN, through a FritzBox access point, towards a Raspberry Pi 3, which acts as a database server. The clients do not store any data: all measured values are sent immediately to the database with a preset periodicity of approximately 10 s. Both servers and clients run on Raspbian OS, while the server-side database is based on MySQL. Each client is programmed through a Python 3 script.

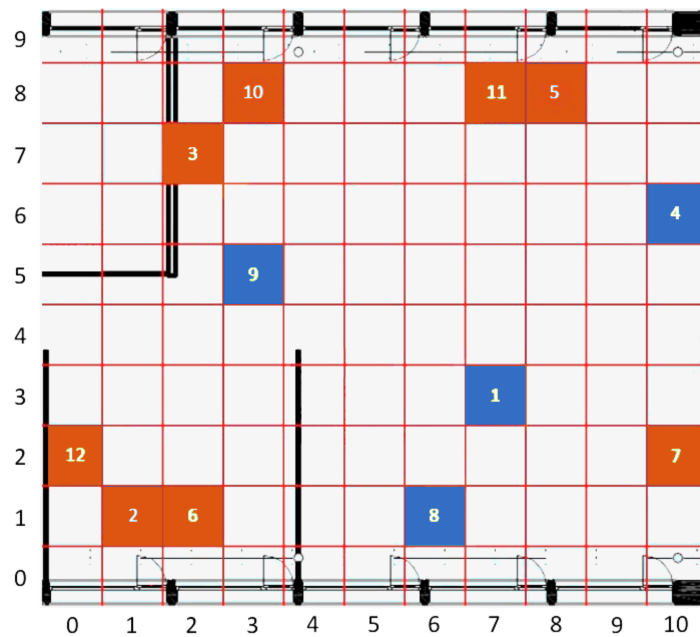


Figure 1. Location of the sensors in the considered premise. The blue cells represent the reference sensors that will be selected during the sensors selection phase.



Figure 2. One of the Raspberry Pi Zero boards used in the work.

The placement of sensors has been organized with the goal of monitoring as large a variety of operating conditions as possible. As an example, sensor 9 is placed near the center of the room, and thus ought to be less affected by weather conditions than, for instance, sensors 2, 5, 6, 8, and 10, which are located close to a window. A grid has been superimposed over the map, which allows us to assign to each sensor a unique coordinate, expressed by a tuple (X, Y) , with $0 \leq X \leq 10$ and $0 \leq Y \leq 9$.

The considered dataset consists of measurements taken from the twelve sensors over the period from 23 October 2019 to 3 March 2020. Data are recorded every 10 s, for a total of around 13,500,000 instances. The maximum observed temperature (Celsius) was 48.95, while the minimum was 5.36. This confirms the impact of weather: while, in the first case, the high temperature can be explained by both the sun influence and the temperature of the Raspberry circuitry, in the latter case it is likely to be caused by an open window placed in close proximity to one of the sensors.

4. Descriptive Analysis

The Pearson correlation values among sensor temperatures are depicted in the lower triangular part of Figure 3, and they show that some of them are naturally more correlated than others. This is the case, for instance, with sensors located close to a window in the upper part of the room. Moreover, sensor 9 shows a high correlation with sensor 1, which

is not surprising, as both of them are placed near the center of the room. However, there are also some notable exceptions. As an example, sensor 10 correlates more with sensor 5 than with sensor 3, despite the fact that it is much closer to the latter than to the former. A similar pattern can be observed for the pair of sensors 12 and 1, as opposed to 12 and 2. These phenomena might be explained by the proximity of a window, the heat emitted by desk lights or other kinds of electrical device, or by the presence of some obstacles that may interfere with air flow, or block the light reaching a specific sensor.

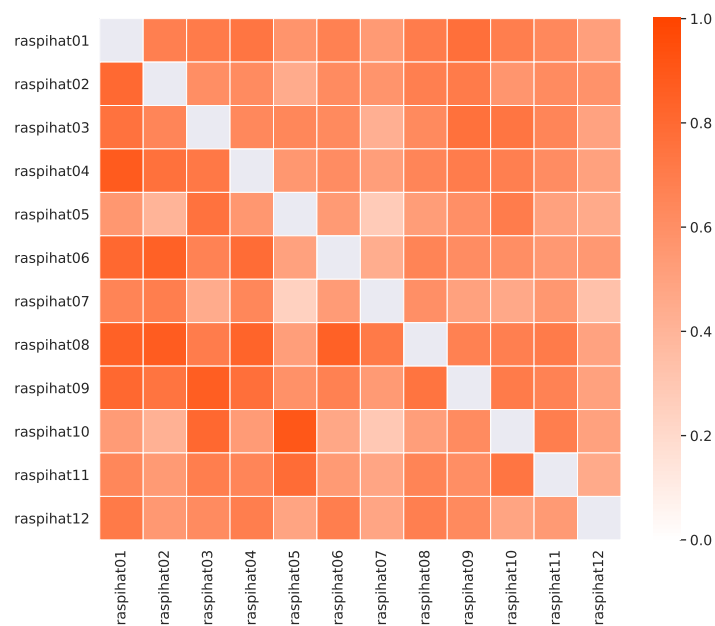


Figure 3. Pearson (lower triangular part) and Kendall (upper triangular part) correlation values among the recorded temperatures.

In order to investigate whether these pairs of neighboring sensors share some non-linear relationships or are simply measuring different phenomena, a further analysis based on Kendall's tau, a non-linear rank correlation measure, has been carried out. Results are depicted in the upper triangular part of Figure 3. In this case, the correlation between the pair of sensors 12 and 2 is higher than the one estimated for the pair of sensors 12 and 1. Conversely, sensor 10 still correlates more with sensor 5 than with sensor 3, which, not being directly adjacent to a window, is probably measuring a different kind of phenomenon. As for the temporal evolution of the temperatures, Figure 4 shows the measurements of three different sensors for the day 25 October 2019, which follow a typical pattern along a 24 h period. It is worth noticing that the temperature recorded by sensor 11 has a spike around 11:30 a.m. in the morning. This is probably due to the heating effect of direct sunlight, that is not present on sensor 8, which is close to another window but on the opposite side of the building. Instead, the latter sensor shows a drop in the temperature at around 5:00 a.m., which can be explained by the daily cleaning staff operations, which include opening the nearby windows to circulate air. Finally, sensor 9 is characterized by a rather stable behaviour, being placed close to the center of the room. It is worth pointing out that, as witnessed by the high average temperatures, sensors have not been calibrated. This was done on purpose, in order to evaluate the performance of the proposed methods in a more challenging scenario, also considering that a proper calibration process might not always be possible in a real-world deployment.

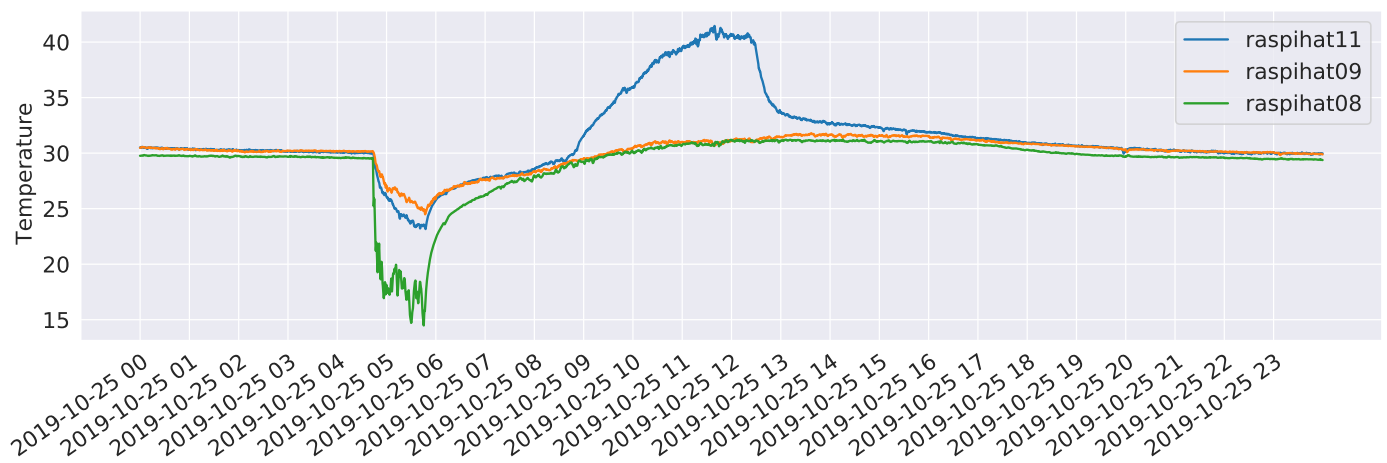


Figure 4. Temperatures recorded by sensors 8, 9, and 11 on the day 25 October 2019.

As we shall see, in the remainder of the work we will always consider an 80–20% training-test split of the dataset, randomly assigning entire weeks to the two partitions. Such a choice has been made taking into account the fact that the recordings span less than one year and cover months that typically exhibit considerable meteorological variations. Thus, it would otherwise have been difficult for the machine learning approaches that we are going to consider to learn models over a subset of the first days, which are capable of generalizing well to the remaining time period. In addition, upon data inspection, it emerged that even contiguous weeks tend to have rather different behaviours.

The performed analysis allows us to conclude that the considered setting is not trivial, as there are some irregularities and local phenomena that influence and differentiate the temperature recordings of the different sensors, making the temperature prediction task quite challenging.

5. Data Pre-Processing

In order to carry out the experiments, it is necessary to define a set of features that can be used as predictors by the machine and deep learning models we are going to develop. To this end, for each temperature measurement, we considered a set of temporal attributes, that are useful to locate the observation in time. They are *sec_from_midnight*, which tracks the number of seconds elapsed from 00:00:00 h, *dow*, a numerical identifier of the day of the week, and *moy*, a numerical identifier of the month of the year. In order to account for time periodicity (and be able to consider, e.g., the fact that 11:59:59 p.m. is close to 00:00:00 a.m.), we encoded each feature by means of two trigonometric transformations

$$\sin(2 * \pi * x / \delta) \text{ and } \cos(2 * \pi * x / \delta),$$

where x represents the original attribute value and δ is the length of the period, e.g., 12 for the attribute *moy*. As a result, six features were obtained: *dow_sin* and *dow_cos*, whose values are shown in Figure 5, *moy_sin* and *moy_cos*, whose values are shown in Figure 6, and *sec_from_midnight_sin* and *sec_from_midnight_cos*, whose values are shown in Figure 7.

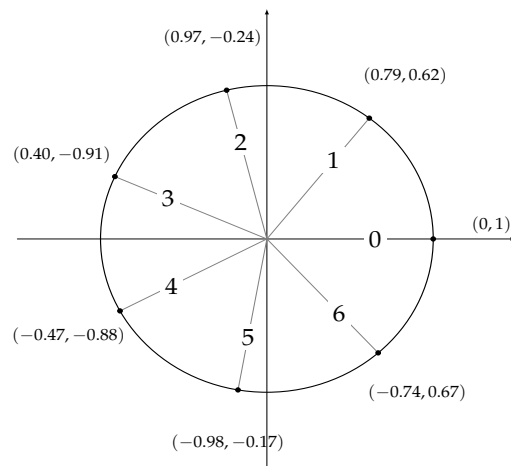


Figure 5. Values assigned to (dow_sin , dow_cos) after a trigonometric transformation of the dow feature, where dow ranges from 0 (Monday) to 6 (Sunday).

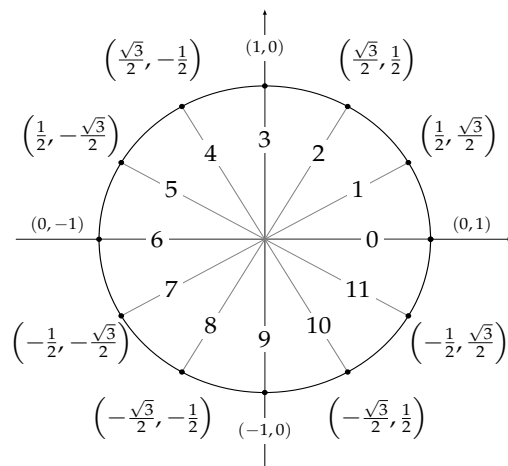


Figure 6. Values assigned to (moy_sin , moy_cos) after a trigonometric transformation of the moy feature, where moy ranges from 0 (January) to 11 (December).

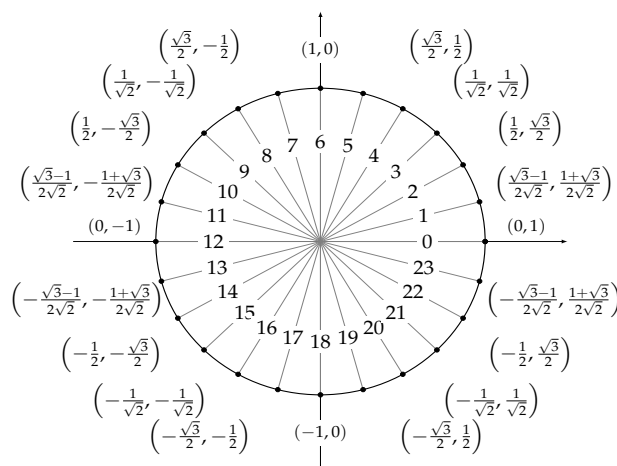


Figure 7. Values assigned to ($sec_from_mid_night_sin$, $sec_from_mid_night_cos$) after a trigonometric transformation of the $sec_from_mid_night$ feature ranging from 0 to 86,399.

5.1. Distance Metrics

The main goal of the work is to study a solution capable of exploiting a subset of sensors to make predictions in place of a sensor to be virtualized. To achieve this, distance metrics can be useful to decide which physical sensors to consider as predictors, and to compute the predicted temperature. Furthermore, distance metrics may be quite useful for the task of optimizing the positioning of a given set of sensors within a certain environment.

The following distance metrics between two sensors, $sensor_i$ and $sensor_j$, whose spatial positions are, respectively, $p_i := (x_i, y_i)$ and $p_j := (x_j, y_j)$, have been considered

- **Euclidean distance:** length of a line segment between the two points p_i, p_j defined as the L_2 -norm $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$;
- **Manhattan distance:** the L_1 -norm of the distance, defined as $|(x_i - x_j)| + |(y_i - y_j)|$;
- **Chebyshev distance:** the L_∞ -norm of the distance, defined as $\max\{|(x_i - x_j)|, |(y_i - y_j)|\}$;
- **Genetic Programming distance:** a combination of the previous three distances obtained by means of a genetic programming algorithm which generates a computation tree whose leaves may contain the three aforementioned distance values or a randomly generated constant and whose internal nodes are the scalar/vector operations defined as a set of primitives;
- **Pearson correlation:** it expresses a possible linear relationship between the statistical variables given by the temperature values of the two sensors, and it is defined as

$$\left(\sum_{k=1}^n (t_i^k - \bar{t}_i) (t_j^k - \bar{t}_j) \right) / \left(\sqrt{\sum_{k=1}^n (t_i^k - \bar{t}_i)^2} \sqrt{\sum_{k=1}^n (t_j^k - \bar{t}_j)^2} \right),$$

where n is the sample size, t_i^k and t_j^k are the individual sample points for $sensor_i$ and $sensor_j$, that is, the temperature values, and \bar{t}_i and \bar{t}_j are the sample means;

- **Kendall correlation:** it expresses a possible ordinal (non-linear) association between the statistical variables given by the temperature values of the two sensors, and it is defined as

$$\frac{2}{n(n-1)} \sum_{k < l} \text{sgn}(t_i^k - t_i^l) \text{sgn}(t_j^k - t_j^l)$$

where n is the sample size, t_i^k and t_i^l are the individual sample points for $sensor_i$ in position k and l , t_j^k and t_j^l are the individual sample points for $sensor_j$ in position k and l , and $\text{sgn}(x)$ is the function $\text{sgn} : \mathbb{R} \rightarrow \{-1, 0, 1\}$ that returns -1 , if $x < 0$, 1 , if $x > 0$, and 0 otherwise;

- **SHAP distance:** SHAP is a game-theoretic method that allows one to evaluate the contributions to the final result of the different predictors used in a machine learning model, with the relevance of the contribution of a predictor to the model being proportional to its SHAP value [26]. In our case, SHAP values for a generic sensor $sensor_i$ are obtained from an XGBoostRegressor model [27] that predicts the temperature value of $sensor_i$ on the basis of the temperature values of the other sensors. It is worth noticing that such a metric is not symmetric;
- **SAX-CBD distance:** SAX-CBD is a Compression-Based Dissimilarity measure [28] based on the assumption that the size of the compressed file of the concatenation of two discrete time series is inversely proportional to the number of patterns that they share. As a preliminary step, the temperature values obtained from $sensor_i$ and $sensor_j$ are discretized by means of Symbolic Aggregate approXimation (SAX) [29]; then, the value of the distance is computed as

$$\text{size_of}(\text{compress}(d_i ++ d_j)) / (\text{size_of}(\text{compress}(d_i)) + \text{size_of}(\text{compress}(d_j))),$$

where $++$ is the concatenation operator, d_i and d_j are the time series consisting of, respectively, the discrete temperature values related to $sensor_i$ and $sensor_j$, and $compress$ (data) are the output of the application of the algorithm DEFLATE [30] to $data$.

The estimation of the above metrics is straightforward and it has been made considering only the training set data, in order to obtain an ordering of the remaining 11 sensors, from the closest to the farthest one. These ranks are calculated for each distance metric listed above.

The genetic programming algorithm was designed relying on the Distributed Evolutionary Algorithms in Python (DEAP) framework [31], considering 100 generations with a population of 600 individuals, i.e., computation-tree-encoded functions. As for the evolutionary operators, we employed one point crossover, with a probability equal to 0.7, and a mutation, with probability 0.4, where a randomly chosen primitive from an individual is replaced by another randomly chosen operation within the primitive set. The chosen selection method is the double tournament [32], which evaluates both the fitness and the size of the individuals in order to discriminate good solutions, following a three-individuals fitness-based first tournament and a size-based second tournament with a parsimony size of 1.4. This last tournament favours the choice of low-complexity solutions, represented by trees of limited height. Individuals are built considering the three (0–1 normalized) distance metrics *Euclidean distance*, *Manhattan distance*, and *Chebyshev distance*, and a set of random constants ranging from -1 to 1 as terminal leaves. The set of primitives consists of the following scalar/vector operations: \min , \max , $+$, $-$, $*$, \div , \log_{10} , *exponentiation*, *square_root*, *negation*, and *absolute_value*. Note that, despite the normalization step performed on the distance metrics, the *absolute_value* operation is still useful given the presence of potentially negative constants in the tree.

In order to determine the fitness function, we considered a further 90–10% training-validation subsplit of the training data, randomly assigning entire weeks to the two sets. In both datasets, each instance consists of a label, that is, the temperature recording of a given sensor, and a list of predictors, which are the 11 temperature values of the other sensors, and the temporal features *sec_from_midnight_sin*, *sec_from_midnight_cos*, *dow_sin*, *dow_cos*, *moy_sin*, and *moy_cos*. The fitness function was computed for each individual as follows: we assessed the prediction error obtained from a series of linear regression models built on the training split and evaluated on the validation split; different models were trained and evaluated, considering each different sensor as a target and increasingly discarding other predictor sensors according to the rank defined by the distance function encoded by the individual, starting from the sensor with the highest value. Then, for each number of considered predictors, we summed the resulting prediction errors, coming from the different target sensors, obtaining an error curve. Finally, to determine the fitness value, we calculated the area under the curve. The computation tree generated by the genetic programming algorithm is shown in Figure 8. It is equivalent to the function $GP_function(d_1, d_2, d_3) = (|d_2|^{d_1} / \log_{10}(d_3)) * c_1$, where d_1 is the *Euclidean distance*, d_2 is the *Manhattan distance*, d_3 is the *Chebyshev distance*, and $c_1 \simeq 0.817$.

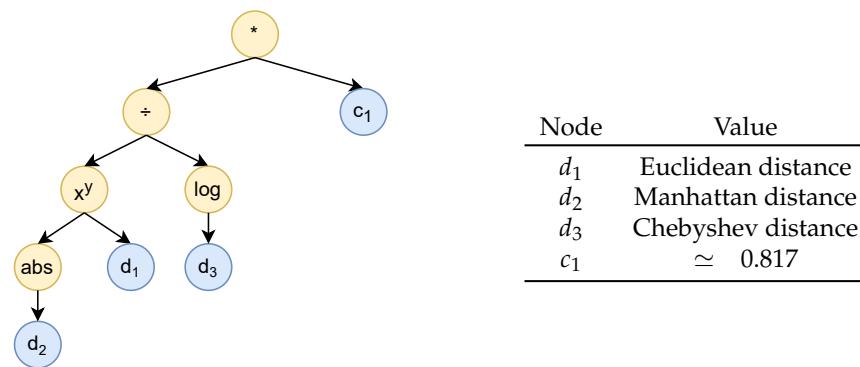


Figure 8. The computation tree generated by the genetic programming algorithm.

The evaluation of the rankings generated by each distance metric d was carried out according to the following procedure on the 80–20% training-test split:

- For each sensor $sensor_i$, the rank $rank_{d,i}$ of the other sensors according to the metric d was considered;
- then, we proceeded in an iterative way: for $k \in \{0, \dots, 10\}$, k sensors among the worst ones in $rank_{d,i}$ were discarded and a regression model was built. In more detail,
 - The sensors whose temperature values were to be used as predictors were determined considering the set of all sensors, except $sensor_i$ and the k sensors located in the last k positions of $rank_{d,i}$;
 - Exploiting the training set data, a linear regression model was built to predict the temperature of $sensor_i$ using as input the temperature values of the sensors selected in the previous step and the features moy_sin , moy_cos , dow_sin , dow_cos , $seconds_from_midnight_sin$, and $seconds_from_midnight_cos$;
 - The resulting model was evaluated on the test set.

For each value of k , the sum of the 95th percentile of the absolute errors obtained for each predicted sensor was computed, obtaining a curve over k . Then, the final error for each metric was determined by calculating the area under the considered curve.

The outcome of the evaluation is reported in Figure 9. The curves show a bowl-shaped pattern. This can be explained by the fact that the first discarded sensors may have little correlation with the temperatures to predict, and thus they may interfere with the accuracy of the final result. On the other hand, the lastly discarded ones were probably carrying useful information. The best metrics turned out to be *genetic programming distance* and *Pearson correlation*.

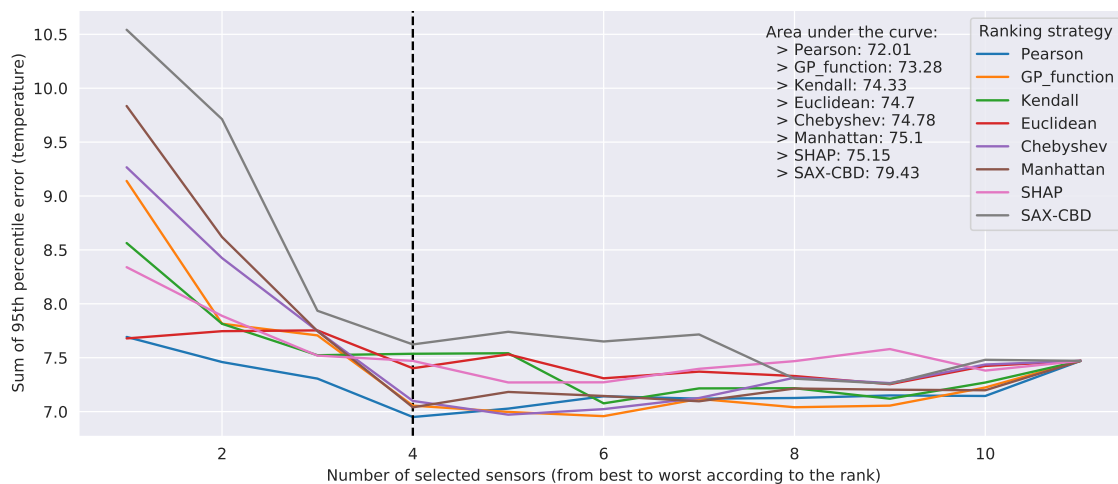


Figure 9. Performance of linear regression, evaluated discarding sensors based on training set ranks. The dashed vertical line represents the elbow of the *Pearson* error graph.

More precisely, *genetic programming distance* provided the best results when considering a subset of at least five sensors as predictors and, as expected, generally outperformed all the other metrics based exclusively on spatial distances. The *Pearson correlation* showed an overall better performance than *genetic programming distance*, in particular when four sensors or fewer were considered as predictors. However, *Pearson correlation* cannot be employed in a more general setting, that is, to determine the best sensors for use as features to predict the temperature at a given cell, where a reference physical sensor may or may not be present. As for the *genetic programming distance*, while we cannot exclude its ability to generalize on each cell, the latter should be actually demonstrated, and it will be the subject of future investigations based on physical data simulations. As a last remark, we observe that Figure 9 clearly shows that excellent results, in terms of prediction error, can be obtained from just a subset of 4–6 sensors, while they become worse when considering three sensors or fewer.

5.2. Sensors Selection

In order to automatically select a subset of sensors to be used as predictors, we need to specify a procedure to determine the number n_{refs} of sensors to select and to establish which sensors to actually consider. To this end, we make use of a procedure based on the Borda count voting method [33]. As a preliminary step, let us introduce some auxiliary notions. First, for $i \in \{1, \dots, 12\}$, let $rank_i$ be the ranking of sensor i obtained by sorting in descending order the remaining 11 sensors according to their Pearson correlation with reference to sensor i . Then, let us define w_i , with $i \in \{1, \dots, 12\}$, as the weight of $rank_i$, defined as $1 - (\varepsilon_i / \max_{i \in \{1, \dots, 12\}} \varepsilon_i)$, where ε_i is given by the sum of the 95th percentile of the absolute errors evaluated for $rank_i$, computed training different LinearRegression models, varying the number of sensors used as predictors from 1 to 11, as described for the case of the metric evaluation procedure at the end of the previous section (Section 5.1).

The procedure consists of the following five steps: (i) for each sensor i , we compute $rank_i$; (ii) for each sensor j , we determine its weighted Borda count, which is defined as $vote_j = \sum_{i \in \{1, \dots, 12\}, i \neq j} (n_{sensors} - pos_i(j)) \cdot w_i$, where $pos_i(j)$ is the position of sensor j in $rank_i$, $n_{sensors} = 12$ is the total number of used sensors, and w_i is the weight of $rank_i$; (iii) the sensors are sorted in descending order according to their final weighted Borda count vote; (iv) an approximation of the elbow of the curve obtained in the previous step is computed by using the Kneedle algorithm [34]—the x -axis value corresponding to the elbow represents the point of maximum curvature of the graph, and the best trade-off between prediction accuracy and number of sensors, after which a law of diminishing returns applies: we choose it to be the n_{refs} value which, in our case, corresponds to four reference sensors (dashed line in Figure 10); (v) finally, the first n_{refs} sensors are selected as the reference ones. It is worth underlining that the weights w_i in the vote-counting formula allow us to offer greater importance to the ranks that provided better results with respect to the error obtained on the validation data split. In addition, the sensors located on the left side of elbow intuitively correspond, by construction, to the best predictors.

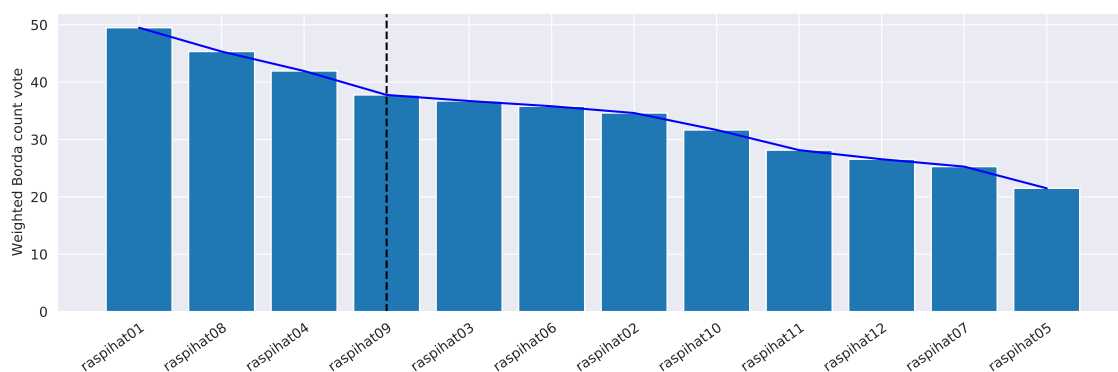


Figure 10. Weighted Borda count vote for each sensor. The vertical line represents the elbow of the graph, and it separates the selected sensors from the discarded ones.

Figure 10 shows the weighted Borda count votes for all sensors obtained by applying the procedure to the considered setting. The proposed criterion led to the selection of sensors 1, 8, 4, and 9 as the references for the prediction. It is noteworthy that the elbow estimate also corresponds to the minimum error point of *Pearson distance* in Figure 9. Even if, in our case, no ex aequo placements occur, as a methodology to deal with them, we suggest to consider, for each sensor, the median of its Pearson correlations with respect to the other ones, prioritizing those with higher values.

5.3. Feature Selection

On the basis of the feature engineering and sensor selection phases, we identified 16 attributes that describe each observation: the temporal features *sec_from_midnight_sin*, *sec_from_midnight_cos*, *dow_sin*, *dow_cos*, *moy_sin*, *moy_cos*, the spatial features *01_ref_dist*, *04_ref_dist*, *08_ref_dist*, *09_ref_dist*, and the reference temperatures *01_ref_temp*, *04_ref_temp*, *08_ref_temp*, *09_ref_temp*. To them, we added *X_coord* and *Y_coord*, that is, the two grid coordinates of the sensor that recorded the observation.

At this point, it is necessary to establish which spatial distance metric to consider among *Euclidean distance*, *Manhattan distance*, *Chebyshev distance*, and their combination, obtained via *genetic programming*, for the estimate of *01_ref_dist*, *04_ref_dist*, *08_ref_dist*, and *09_ref_dist*. To this end, we determined the 95th percentile of the validation error values obtained from 5 global 16-attribute XGBoostRegressor models, one for each spatial distance metric, trained on the usual 90% subsplit of the original training data pertaining to all the sensors, except for the reference ones (which are already used as predictors). The outcome of such an analysis is depicted in Figure 11 and led to the selection of the *genetic programming distance*, which outperformed all the other ones. At the end, the following 16 attributes were chosen to describe each observation: the 6 temporal features, the 4 reference temperatures, the 4 spatial features *01_ref_gpdist*, *04_ref_gpdist*, *08_ref_gpdist*, and *09_ref_gpdist*, which are the genetic programming distances from the reference sensors, and the 2 grid coordinates *X_coord* and *Y_coord* of the sensor that recorded the observation.

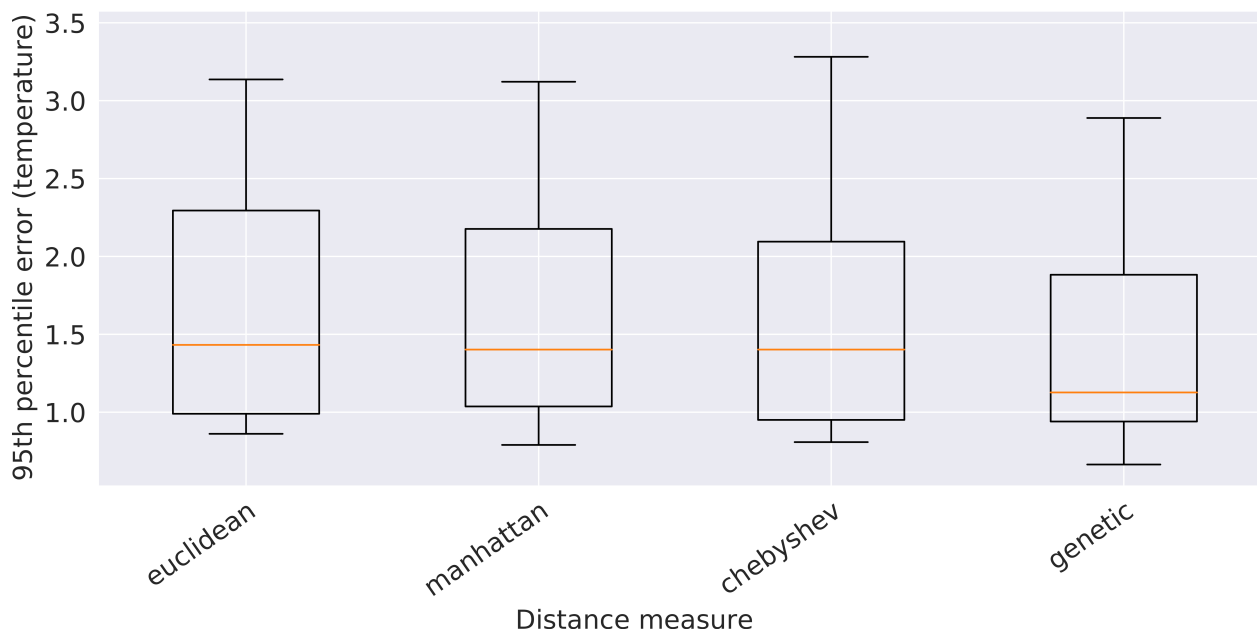


Figure 11. Boxplots of the 95th percentile of the error provided by the XGBoost models built on the 5 spatial distances considered in this work.

Since some of these attributes may be redundant, we executed a two-step feature selection process working on the training split. As a preliminary data preparation step, all attributes were standardized by subtracting their mean and dividing by their standard

deviation. The first selection step searched for highly correlated attributes, that is, attributes with a Pearson correlation value above 99%. As a matter of fact, no feature was removed from the dataset by this step. The second step evaluated the potential impact of the remaining attributes on the final prediction. To this end, the SHAP values extracted from a single global XGBoostRegressor model trained on the training data split related to all the sensors, except for the reference ones, were taken into consideration. The outcome of such an analysis is depicted in Figure 12. We first observe that all reference temperature values have a large impact on the final prediction and, naturally, higher values of these attributes increase the value of the prediction. Focusing on the office map, there is a clear variation across the X and Y axes. As for the Y axis, from the distribution of the values, we can conclude that, during the data collection period, the northern side of the room was generally warmer than the southern one. As for the X axis, the eastern side of the room seems to be warmer than the western one. Interestingly, the *moy_cos* and *dow_cos* features do not seem to be important for the overall prediction when compared to the counterpart obtained from the sine transformation. As pointed out by Figures 5 and 6, this means that the contribution to the prediction given by the features that discriminate the first half of the week/year from the second half are more important than those that discriminate the first/fourth quarters from the second/third ones. Furthermore, the genetic programming distances from sensors 1, 4, and 9 are considered of marginal importance when compared to the distance from sensor 8. On the basis of the SHAP results, we ultimately decided to remove the five attributes *dow_cos*, *moy_cos*, *01_ref_gpdist*, *04_ref_gpdist*, and *09_ref_gpdist*, ending up with a total of eleven attributes.

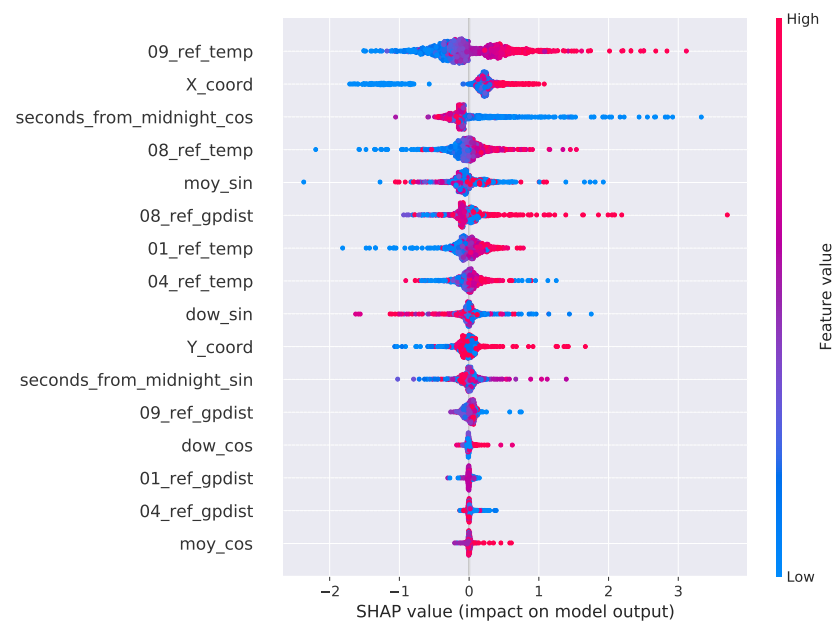


Figure 12. SHAP values of the attributes considered in the second step of the feature selection process.

From a general point of view, the first correlation-based feature selection step must be considered as a preliminary, coarse screening of the predictor variables; it should discard a feature when it is found to be almost identical to another one in the dataset, without the risk of removing predictors that might still be preserved by the subsequent, SHAP-based feature selection step. This is the reason that we recommend relying on a high threshold, which should nevertheless be established considering the specific scenario. Indeed, such a pre-screening may be useful in a more general situation, characterized by a large amount of input features, to reduce the time requirements needed to train the XGBoost model in the second step of the feature-selection phase. In our case, as already discussed, relying on a correlation threshold of 99% led to no attribute being discarded. However, lowering the threshold to 94% would have led to the removal of feature *04_ref_gpdist*, which would also

have been discarded in the subsequent SHAP-based step. To remove the first feature kept in the SHAP-based step, namely *04_ref_temp*, the correlation threshold should have been reduced to a value less than or equal to 88%.

6. Predictive Analysis

To determine the performance of the sensor and feature selection phases with respect to the task of temperature virtual sensing, and to identify the best prediction methodology, we experiment with and contrast the following approaches:

- Baseline: simple and Inverse Distance Weighted (IDW) average of the temperatures;
- Particle filters;
- LinearRegression – Python’s package Scikit-learn [35];
- XGBoostRegressor – Python’s package xgboost [27];
- An LSTM recurrent neural network, trained by means of the PyTorch Deep Learning library [36].

As pointed out in Section 5.2, except for the baseline methods and particle filters, predictions make use of the temperature values recorded by the four chosen reference sensors (sensors 1, 4, 8, and 9) depicted in blue in Figure 1. To ensure the comparability of the results obtained from the various approaches, prediction errors were evaluated on the remaining eight sensors (the original 12, except the four reference sensors that are already used as predictors). In addition, as previously mentioned, we always considered the same 80–20% split training-test of the dataset.

The predictive analysis tasks are organized as follows. In Sections 6.1–6.4 we evaluate the various approaches to temperature virtual sensing. Then, we analyze the prediction errors: Section 6.5 assesses the uncertainty associated with the predicted quantities, Section 6.6 discusses the errors per single sensor, and Section 6.7 links the prediction error to the available training data. Finally, in Section 6.8 the outlined framework is evaluated with respect to the optimal result that can be achieved by means of a brute force approach to sensor selection.

6.1. Baseline Methods

This first analysis allowed us to define a baseline against which to compare the results of the other approaches. Given a sensor for which we want to predict the temperature readings on the test set, the idea is that of approximating such values by a simple combination of the temperatures recorded by the other ones at the same time instant. To this end, we applied two different techniques: classical average and Inverse Distance Weighted Average (IDWA), according to which closer sensors have an impact on the overall prediction greater than that of sensors which are farther away. In more detail, the weight assigned to the i -th predictor is computed as $w_i = \frac{1}{d(x, x_i)}$, where $d(\cdot, \cdot)$ is the genetic-programming-based distance (Section 5.1) between two points of the grid, x_i is the position of the i -th sensor, and x is the position of the sensor to predict. The weighted temperatures are summed and then divided by the sum of the weights.

Moreover, to determine the impact on the prediction accuracy of the distance between sensors, we performed several experiments by considering as predictors just the k sensors closest to the one to predict, for $k \in [1, 11]$. For each approach and sensor to evaluate, we determined the temperature absolute error, considering its 95th percentile, ϵ_{95} , which can be thought of as a worst-case prediction scenario. Figure 13 collects the boxplots of ϵ_{95} . Each boxplot includes a value for every test tensor, for a total of eight values. The orange line represents the median of ϵ_{95} , while the whiskers correspond respectively to the minimum and maximum values excluding the outliers (< 1 st quartile - 1.5 IQR or > 3 rd quartile + 1.5 IQR, where IQR is the interquartile range given by the difference between the 3rd and the 1st quartile). It clearly emerges that classical average is largely influenced by the number of closest sensors used for the prediction. Here, the optimal number of sensors seems to lie in the range [3, 5]. On the other hand, IDWA seems to be less affected by the number of predictor sensors. Indeed, looking at the median, large values of k led

to better results. This is to be expected, as the contributions of the different sensors are already weighted according to their distances.

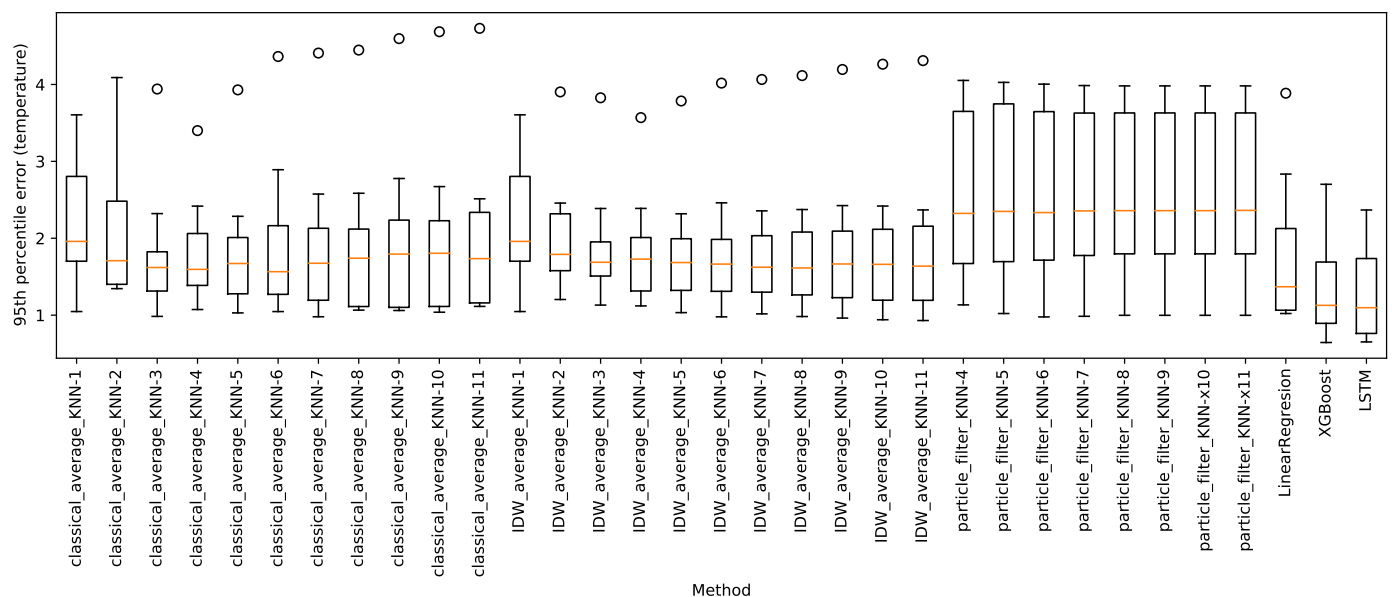


Figure 13. Boxplots of the 95th percentile of the error provided by the considered approaches.

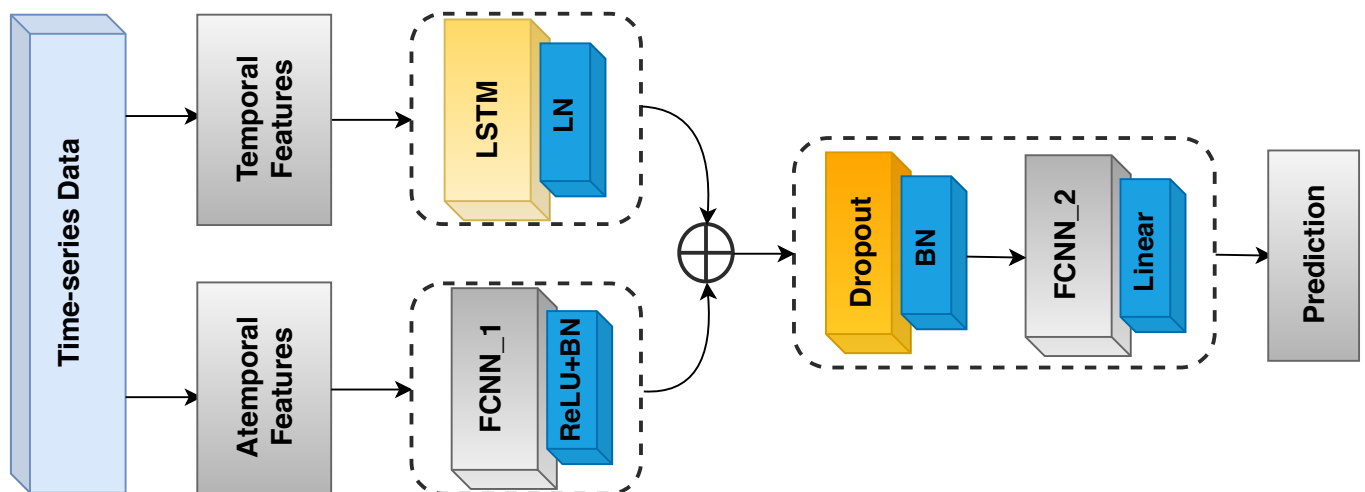


Figure 14. The LSTM recurrent neural network model architecture.

6.2. Particle Filters

Particle filters are a class of Sequential Monte Carlo algorithms used to approximate the internal states of dynamic systems starting from partial measurements with random disturbances, which afflict the sensors as well as the dynamic system itself [5]. Given the noisy and partial observations, this approach aims at measuring the state posterior distributions of some Markov process. Particle filters leverage a set of particles to represent such a posterior distribution. Each particle has an assigned weight, indicating the chance of that particle being sampled from the probability density function of the quantity we want to compute. As for this experimentation, we considered one particle filter for each of the eight evaluated sensors, and the k -closest sensors as landmarks, with $k \in [1, 11]$. Each particle thus represents a likelihood estimation of a temperature and is moved at each time-step following the average temperatures of the landmarks. The likelihood

probability is computed on the basis of the genetic programming distance between particles and landmarks.

As with the baseline approaches, we estimated the error's 95th percentile for each evaluation sensor and each potential k value. The results are reported in Figure 13. It is evident that particle filters are also affected by the choice of k . For the sake of readability, we decided to ignore the results for $k \in [1, 3]$, as those values led to very weak predictions, with boxplot whiskers extending over six degrees. As for the remaining values, according to the median, the best predictions are achieved for values of k equal to 4 or 5.

6.3. Machine Learning Approaches

We considered two different machine learning approaches, namely, a simple Scikit-learn's LinearRegression model, and a more complex XGBoostRegressor ensemble approach.

LinearRegression implements an ordinary least squares linear regression. For the sake of our study, it has been trained on the 11 (standardized) features selected in Section 5, with training labels corresponding to the temperatures recorded by eight evaluation sensors (the original 12, except the four reference sensors that are used as predictors).

XGBoost [27] implements gradient-boosted decision trees focusing on computational speed and model performance. Gradient boosting iteratively builds new models to predict the residuals of errors of previous models exploiting a gradient descent algorithm to minimize the loss [37]. The resulting models are then combined to generate the final prediction. As a first step, we tuned the XGBoostRegressor model with the above-described training set, from which a validation set of size 20%, consisting of randomly chosen weeks, was extracted. The task was performed by means of *Hyperopt* [38], a library for hyperparameter optimization written in Python, minimizing the 95th percentile of the error loss function for 40 evaluation steps on the following hyperparameters: *max_depth*, *learning_rate*, *n_estimators*, *reg_alpha*, *reg_lambda*, *gamma*, *subsample*, *colsample_bytree*, and *min_child_weight*. The resulting values are listed in Table 1. With the tuned hyperparameters, the model was trained on the entire training set, and then evaluated on the test set over the usual eight sensors.

Table 1. XGBoostRegressor parameters (rounded to the 5th decimal digit).

Parameter Name	<i>max_depth</i>	<i>learning_rate</i>	<i>n_estimators</i>	<i>reg_alpha</i>	<i>reg_lambda</i>	<i>gamma</i>	<i>subsample</i>	<i>colsample_bytree</i>	<i>min_child_weight</i>
Value	16	0.015	350	78.87396	0.50044	5.95353	0.66425	0.65694	1.0

The outcomes shown in Figure 13 suggest that the tested machine learning methods vastly outperform the baseline approaches. Specifically, XGBoost shows a better performance than LinearRegression on all 95th error quantiles. Furthermore, XGBoost's boxplot is wider than that of LinearRegression, suggesting a more unstable behaviour across the sensors predictions. This is to be expected, being XGBoost a far more complex and flexible model.

6.4. Deep Learning Approach

Up to this point, to predict the temperature at a given time instant, we considered reference sensor values from the same instant. In the literature on remote and virtual sensing, it has already been shown that deep learning methods are capable of taking temporal and spatio-temporal knowledge into account (see, e.g., [11–13]). Specifically, in our context, it may be the case that the recent history of temperatures reported by the reference sensors provides information that is relevant to the overall prediction. As an example, when opening a window in winter time, one may notice a regular and continuous decrease in the temperatures recorded by a nearby temperature sensor. This, together with information recorded by the other reference sensors, may give the model a hint regarding the temperature propagation in the room. We designed an LSTM-based model that takes

such histories into account. Its architecture, which is depicted in Figure 14, consists of three subparts:

- The first (*temporal*) part (LSTM on the upper left side of Figure 14) takes a history of the four (standardized) reference temperatures as input. Then, a unidirectional LSTM layer, consisting of 128 units, from which we retrieve just the last outputs, followed by LayerNormalization, is applied;
- The second (*atemporal*) part (FCNN_1 on the bottom left side of Figure 14) takes the seven remaining (standardized) attributes as input, resulting from the feature selection process (Section 5). These attributes do not have any significant history, but are still important to generate the final output, since they allow the model to pinpoint the prediction in space and time. The aforementioned seven features are passed to a Dense layer, consisting of 64 neurons, followed by a ReLU activation function and a BatchNormalization layer;
- The third part (FCNN_2 on the right side of Figure 14) takes the outputs of the first two parts and concatenates them, generating a tensor of size 192. Then, BatchNormalization and Dropout with 0.1 rate are applied to the result of such a concatenation. Next, the data go through a dense layer of 128 units, followed by a ReLU activation function, and a BatchNormalization layer. The final output is produced by a single-unit Dense layer with linear activation function.

In order to train the neural network, we relied on Adam optimizer with a 9×10^{-5} learning rate minimizing the mean_squared_error loss function. The architecture of the model and all the other hyperparameters were chosen through iterative random search tuning performed on a fixed 80–20% training-validation subsplit, as was already done for the previous machine learning approaches. At last, the same tuning process suggested a length of 18 samples (equivalent to a period of 3 min) for the reference temperature histories, over a tested range of from 1 to 5 min.

As shown in Figure 13, with respect to the test set, the network essentially provided the same performance as XGBoostRegressor: while the upper whisker is marginally better than that of the ensemble approach, the lower one is slightly worse. Furthermore, the broader extension of the boxplot suggests a more variable behaviour than XGBoost. Based on these results, perhaps surprisingly, we can infer that, in our setting, historical knowledge of temperatures alone does not contribute much to the accuracy of the prediction. This can be justified by the fact that the sensors we want to predict are typically located much closer to a window than the reference ones, and thus they should also be the first to be affected by any weather-related phenomenon. Accordingly, reference sensors' historical data are not so important from this point of view. As an additional confirmation, results of an auto-correlation analysis showed that information conveyed by the reference sensors at the time instant in which the prediction is carried out is way more relevant than information present in the historical temperature values related to the same sensors.

In future work, we plan to evaluate the performance of a CNN-LSTM-based neural network that, in principle, should allow us to relate the temporal dimension of temperature histories to spatial information about the placement of the reference sensors, the distances among them, and their distance from the location we want to predict.

6.5. Prediction Intervals Analysis

Usually, a regression model generates a single value for each prediction, which represents itself a random variable. However, under several circumstances, quantifying the uncertainty associated with the prediction, instead of computing just a single value, is very useful, as it gives an indication of the reliability of the results. This can be done by setting proper prediction intervals. Such intervals provide probabilistic upper and lower bounds on the estimate of an outcome variable and can be computed via quantile regression [39]. Typically, regressions minimize the mean squared-error loss function $L(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$, while quantile regression aims at estimating conditional quantiles of the response variable. This is achieved by adopting the loss function

$L_\gamma(y, \hat{y}) = \sum_{i=y_i < \hat{y}_i}^N (\gamma - 1) |y_i - \hat{y}_i| + \sum_{i=y_i \geq \hat{y}_i}^N (\gamma) |y_i - \hat{y}_i|$, where N is the number of the samples in the training set, γ is the quantile of the response variable to forecast, \hat{y}_i is the predicted value for the i -th sample, and y_i is the real target value for the i -th sample.

We explored two different approaches to quantile regression: a linear regression model and a gradient-boosting regression model [37]. In both cases, two models were trained: one for the upper ($\gamma = 0.025$) and one for the lower ($\gamma = 0.975$) bound of the interval. This means that 95% of the actual values should lie between these two predicted bounds. The training procedure was the same as in Section 6.3 and an excerpt of the results for the sensor 3 test data is shown in Figure 15, in the case of linear regression, and in Figure 16, in the case of gradient-boosting regression. Although both approaches were indeed observed to guarantee that 95% of the observed values end up between their estimated lower and upper bounds, the intervals obtained from the Gradient Boosting models are generally less coarse, thus providing a better approximation of the uncertainty intervals.

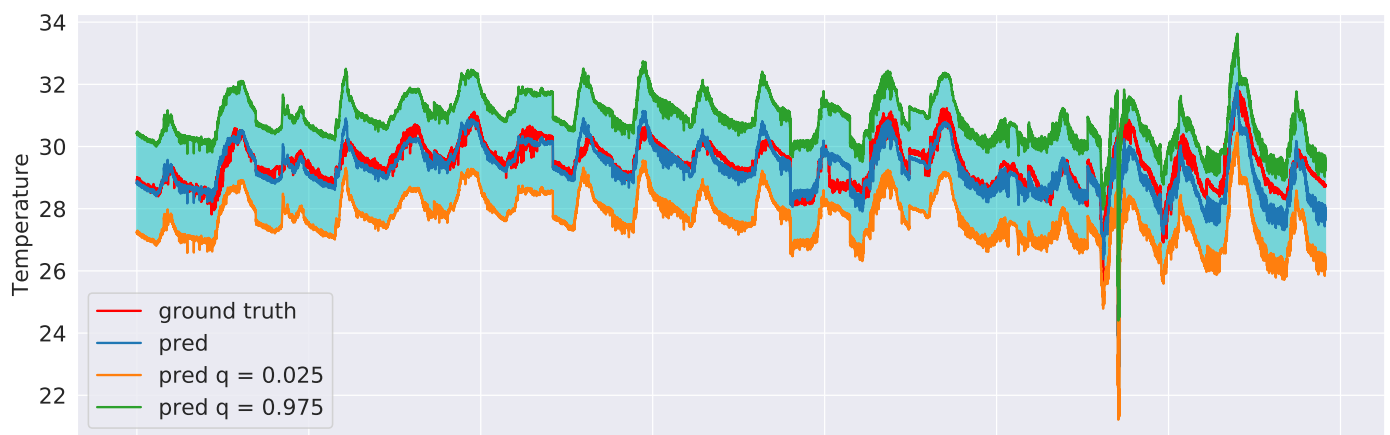


Figure 15. Linear regression prediction intervals related to sensor 3 test data with $\gamma = 0.025$ and $\gamma = 0.975$.

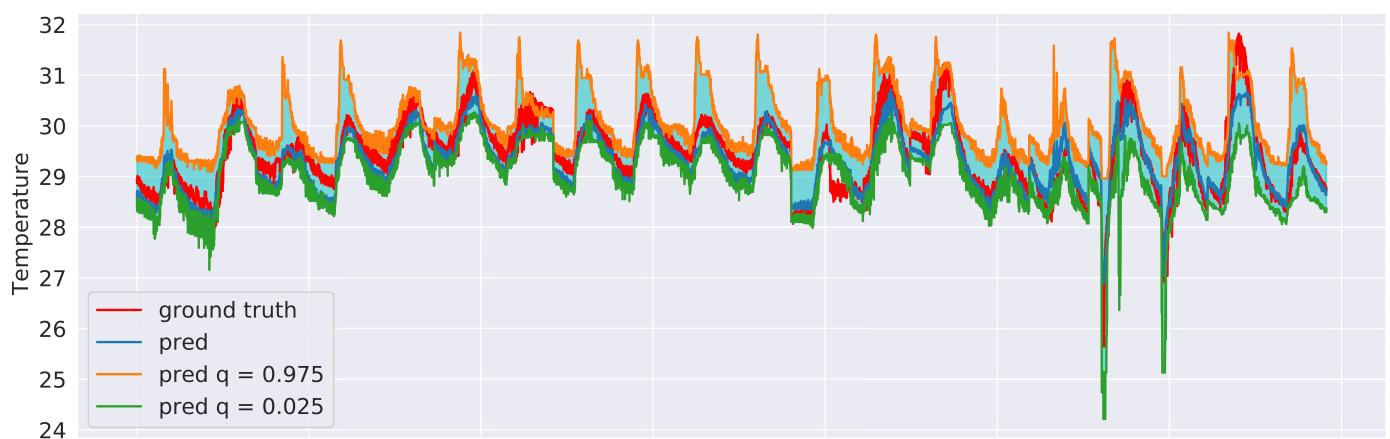


Figure 16. Gradient boosting regression prediction intervals related to sensor 3 test data with $\gamma = 0.025$ and $\gamma = 0.975$.

6.6. Errors per Single Sensor

Let us now take a closer look at the performance of the machine and deep learning approaches. Figure 17 reports the 95th percentile errors of such models for each of the eight evaluation sensors. Although the LSTM and XGBoost models typically exhibit a better performance than LinearRegression, the latter, despite being a much simpler model, is a close match, with the notable exception of sensor 10, where it is vastly outperformed by its contenders. It is also worth observing the relatively high error rates on sensor 7:

upon closer inspection, this sensor displayed a very strange, fluctuating curve, as shown in Figure 18. It may be difficult to predict this kind of irregular temperature values using a global model for all eight sensors, as is done in this work. Therefore, as a future analysis, we plan to compare the outcomes of the global models with those that can be obtained by building a single model for each sensor to be predicted.

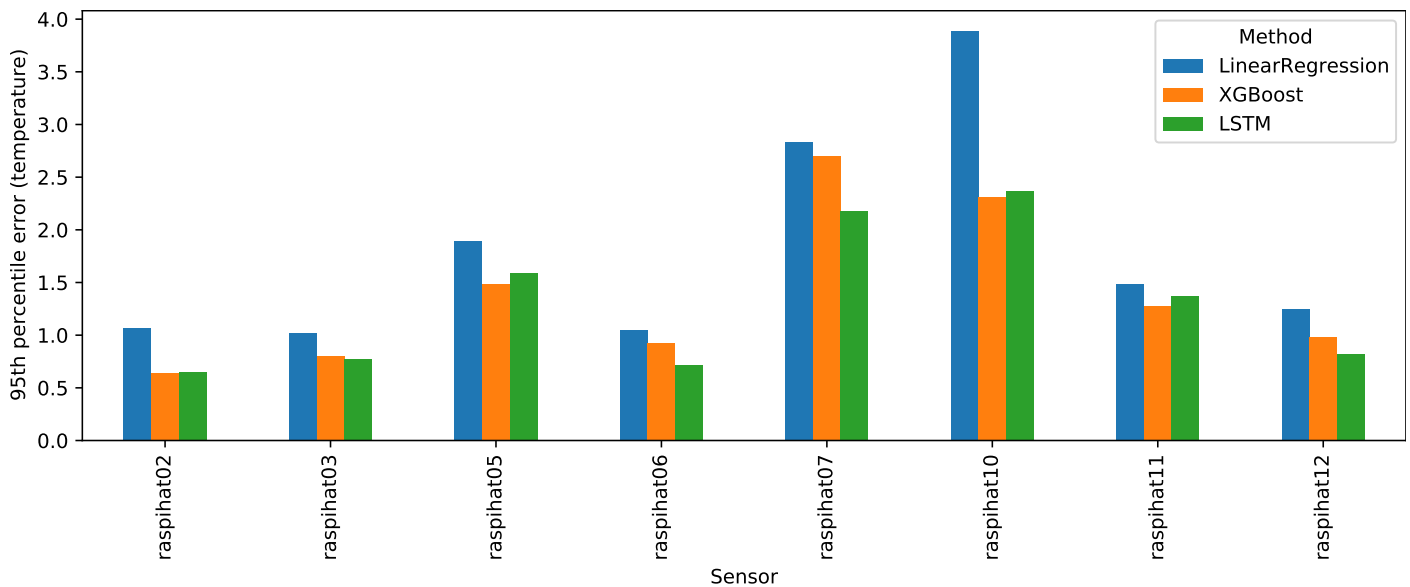


Figure 17. Results of machine and deep learning approaches for each evaluation sensor.

6.7. Effects of Reducing the Training Data Size

Given the large quantity of training data available, we examined the effects of reducing its size, relying on the XGBoost model introduced in Section 6.3. We tested various training set cardinalities obtained by sampling the data at one week granularity. The experiment was conducted using 10 sampling rates of $r \in \{0.1, 0.2, \dots, 1.0\}$, repeating each execution with 10 different random seeds to prevent sampling bias. As expected, the findings in Figure 19 show that, as the training data size decreases, the median of the 95th percentile of the errors raises, probably due to the fact that some seasonal trends may be overlooked if a too-small data sample is employed.



Figure 18. Sensor *raspihat07* temperature values related to the prediction error outliers compared with the three nearest-neighbour sensors.

To further investigate the prediction error related to the outliers, we iteratively discarded the test data belonging to different devices, and it ultimately emerged that the

anomalous values belonged to the predictions made for sensor *raspihat07*. Figure 18 shows the temperature values of sensor *raspihat07* and of its three closest neighbors (*raspihat01*, *raspihat04*, and *raspihat08*) limited to the time instants at which outliers are present in the prediction error (*mean absolute error* ≥ 2 °C). At these time instants, sensor *raspihat07* shows a more marked fluctuation behaviour for both high and low values, which may suggest a degradation or bad calibration of the device. Finally, we reran the training data size reduction experiment, discarding the data related to sensor 7 from the evaluation. In this way, a more predictable monotonic decrease in the prediction performance was observed.

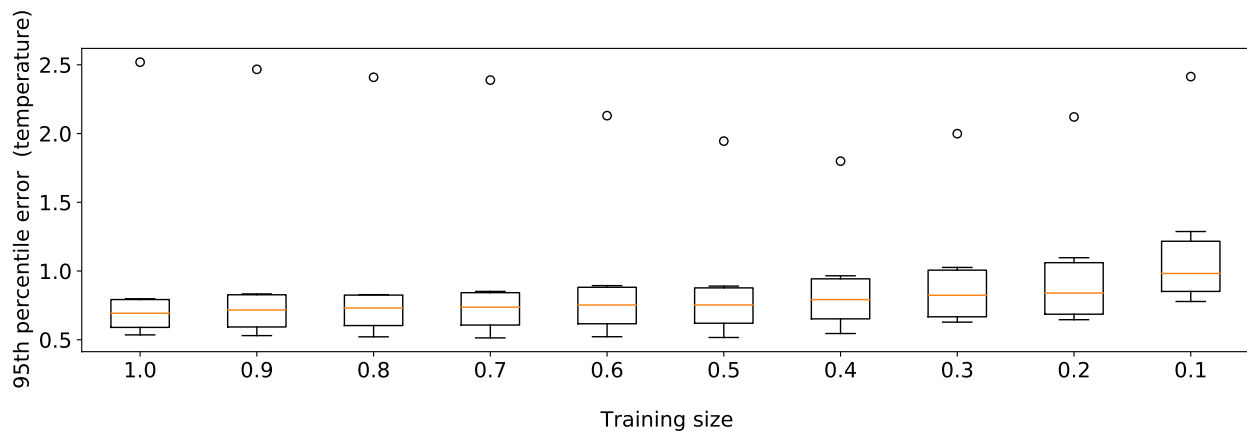


Figure 19. XGBoost results obtained by varying the training set size.

6.8. Comparison with a Brute Force Approach

Last but not least, in order to evaluate the effectiveness of the proposed sensor selection procedure, we carried out a comparison with a brute force approach which considers any possible combination of four sensors or fewer, chosen as the reference ones. For each possible combination, we trained an XGBoost model on the training data split, based on the features discussed in Section 5.3, before the feature selection phase. To ensure comparability, we performed our evaluation considering the test data split of the four sensors 2, 5, 7, and 10, since they cover the whole room and are also typically placed in the lower tier of the rankings analysed in Section 5.1. Excluding the aforementioned four sensors, the possible reference sensor combinations are $\sum_{k \in \{1, \dots, 4\}} \binom{8}{k}$. As for the cardinality k of the subset of selected reference sensors, a maximum value of four was chosen to ensure that there is no combination with a lower number of sensors capable of providing a better prediction accuracy than the one obtained with the proposed solution. Results are shown in Figure 20. The performance of the proposed solution is very close to the optimal one achieved by brute force. This is remarkable, especially considering that the brute force solution performs an exhaustive search in the problem space, and can thus only be applied in scenarios involving a small number of sensors.

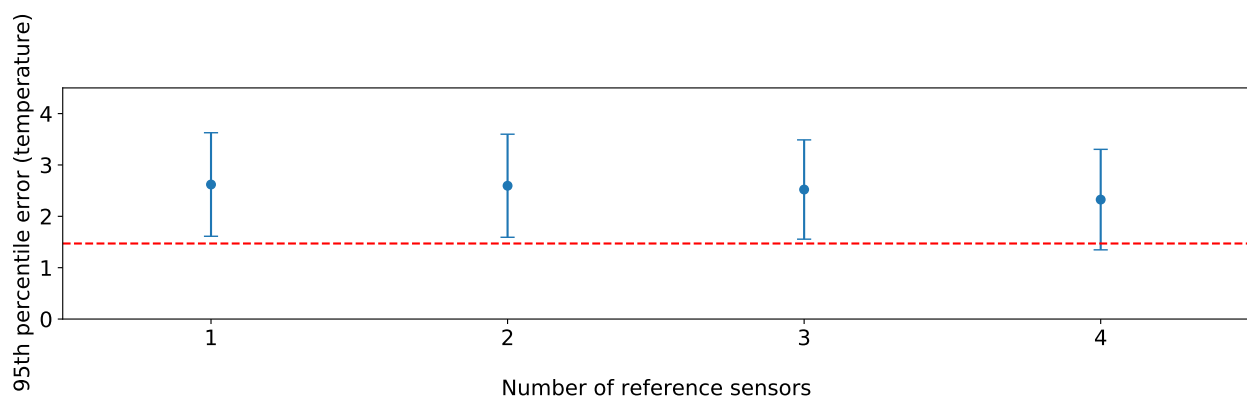


Figure 20. Results obtained from XGBoost on all possible combinations of k reference sensors, with $k \in \{1, \dots, 4\}$. For each value of k , the vertical line represents the extent of the errors given by the different combinations, while the dots represent the median error. The red dashed horizontal line represents the error obtained by the subset of reference sensors selected by our approach.

7. Discussion

In the previous sections, we focused on how to perform virtual sensing efficiently, dealing with the problem from different perspectives. While it is clear that prediction performances are expected to improve, together with the number of sensors and historical training data available, it is typically worth finding a trade-off between the accuracy of the models and the cost of the employed resources.

Our proposed solution encompasses all the relevant aspects of virtual sensing, including sensor selection, the estimation of the needed amount of training data, the choice of predictive model, and the evaluation of its performance. Most importantly, the approach can be regarded as a *black-box*, completely independent from the physical characteristics of the considered scenario, such as any element capable of influencing the internal temperature (windows, radiators, etc.). Thus, in principle, it can be applied to any generic indoor environment with an arbitrary set of pre-located sensors.

Here, we provide a final overview of the overall procedure, whose steps, as portrayed in Figure 21, are as follows: (i) a set of sensors is placed inside a room; (ii) the temperature values measured by these sensors are collected over a significant period of time; (iii) following the steps outlined in Section 5.2, a subset of sensors is selected from those present in the room by exploiting the Pearson correlation of their temperature measurements; (iv) the other sensors can be removed from the room and replaced by the output of predictive models built using information from the remaining reference sensors, as shown in Section 6.

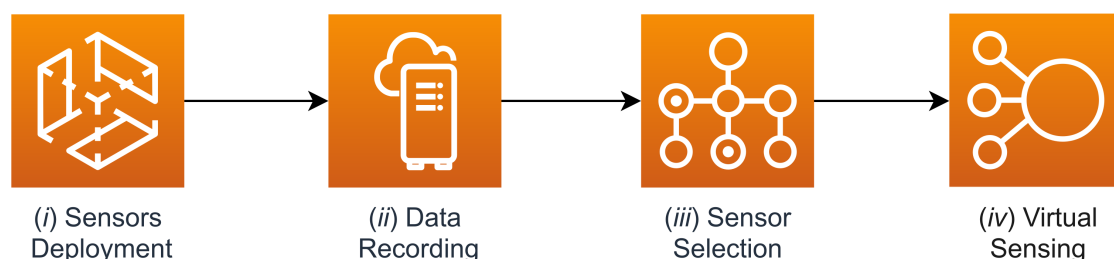


Figure 21. High-level steps which characterize the proposed procedure.

Specifically, it is once again worth highlighting the role played by the most important component of the procedure, i.e, the proposed sensor selection technique, which provided a very good result when compared to a plain brute force approach. This could be achieved leveraging the weighted Borda count method in combination with the Kneedle algorithm:

the former allows us to give a greater importance to the sensor ranks that provided the better results, thus favoring sensors that ought to be the most relevant predictors; the latter reasonably approximates a trade-off point at which the cost of increasing the number of sensors used as predictors is no longer worth the corresponding performance benefit.

8. Conclusions and Future Work

In this paper, a virtual sensing application in the context of an open space office which has been regularly operating for several months has been considered. First, we systematically analysed various distance metrics that can be used to determine the best sensors on which to base a temperature prediction task. This also led to the design, by means of a genetic programming approach, of a combined metric, that can effectively be exploited for virtual sensing. Based on such findings, we proposed an original approach to sensor selection, and then conducted a thorough evaluation of several virtual sensing methodologies for the prediction of temperatures. Results showed that, within the considered setting, it is possible to achieve a satisfactory prediction performance using relatively simple models with a limited subset of predictor sensors in spite of the complex multi-parameter scenario.

As for future work, besides the directions already mentioned in the paper, we look forward to creating models to perform temperature interpolation and temperature trend simulation, and to compare their outcomes with those acquired from a specifically developed physical simulation of the room temperatures. Furthermore, on the basis of such simulation data, we will investigate techniques for the optimal arrangement of a limited number of sensors in a room. The proposed genetic programming-based distance metric is a first step in this direction. Finally, as a parallel research direction, we intend to incorporate some pieces of physical knowledge in the machine/deep learning models, according to a *grey-box* approach.

Author Contributions: Conceptualization, A.M. (Angelo Montanari), F.P., and A.U.; methodology, A.B., F.P. and A.U.; software, A.U. and A.B.; validation, F.P. and A.M. (Angelo Montanari); formal analysis, A.M. (Angelo Montanari); investigation, A.U. and F.P.; resources, F.P.; data curation, A.B.; writing—original draft preparation, A.U.; writing—review and editing, A.B., F.P., A.M. (András Montvay) and A.M. (Angelo Montanari); visualization, A.U.; supervision, A.M. (Angelo Montanari) and (F.P.); project administration, A.M. (András Montvay), A.M. (Angelo Montanari) and F.P.; funding acquisition, A.M. (András Montvay), A.M. (Angelo Montanari) and F.P. All authors have read and agreed to the published version of the manuscript.

Funding: The authors would like to acknowledge the funding received from the Competence Centre ASSIC—Austrian Smart Systems Integration Research Center, co-funded by the Federal Ministries of Transport, Innovation and Technology (BMVIT) and Science, Research and Economy (BMWFW) and the Federal Provinces of Carinthia and Styria within the COMET—Competence Centers for Excellent Technologies Programme. Moreover, the authors would like to thank Google Academic Research Grant and TensorFlow Research Cloud programs for the resources provided on Google Cloud Platform.

Data Availability Statement: Data available in a publicly accessible repository <https://github.com/dslab-uniud>, accessed on 7 April 2021.

Acknowledgments: The authors also thank M. Kraft, V. Dimitrievska, A. Horn, and C. Mayer from Silicon Austria Labs GmbH for the helpful discussions and support.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Liu, L.; Kuo, S.M.; Zhou, M. Virtual Sensing Techniques and their Applications. In Proceedings of the 2009 International Conference on Networking, Sensing and Control (ICNSC), Okayama, Japan, 26–29 March 2009; pp. 31–36.
2. Li, H.; Yu, D.; Braun, J.E. A Review of Virtual Sensing Technology and Application in Building Systems. *HVAC&R Res.* **2011**, *17*, 619–645.
3. Saheba, R.; Rotea, M.; Wasynczuk, O.; Pekarek, S.; Jordan, B. Virtual Thermal Sensing for Electric Machines. *IEEE Control Syst. Mag.* **2010**, *30*, 42–56.

4. Oktavia, E.; Mustika, I.W. Inverse Distance Weighting and Kriging Spatial Interpolation for Data Center Thermal Monitoring. In Proceedings of the 1st International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE), Yogyakarta, Indonesia, 23–24 August 2016; pp. 69–74.
5. Doucet, A.; De Freitas, N.; Gordon, N. An Introduction to Sequential Monte Carlo Methods. In *Sequential Monte Carlo Methods in Practice*; Springer: Berlin, Germany, 2001; pp. 3–14.
6. Wang, J.; Zheng, Y.; Wang, P.; Gao, R.X. A Virtual Sensing based Augmented Particle Filter for Tool Condition Prognosis. *J. Manuf. Process.* **2017**, *28*, 472–478. [[CrossRef](#)]
7. Montzka, C.; Moradkhani, H.; Weihermüller, L.; Franssen, H.J.H. Hydraulic Parameter Estimation by Remotely-sensed Top Soil Moisture Observations with the Particle Filter. *J. Hydrol.* **2011**, *399*, 410–421. [[CrossRef](#)]
8. Wang, J.; Xie, J.; Zhao, R.; Zhang, L.; Duan, L. Multisensory Fusion based Virtual Tool Wear Sensing for Ubiquitous Manufacturing. *Robot. Comput. Integr. Manuf.* **2017**, *45*, 47–58. [[CrossRef](#)]
9. Lary, D.J.; Faruque, F.S.; Malakar, N.; Moore, A. Estimating the Global Abundance of Ground Level Presence of Particulate Matter (PM_{2.5}). *Geospatial. Health* **2014**, 611–630. [[CrossRef](#)]
10. Lary, D.J.; Alavi, A.H.; Gandomi, A.H.; Walker, A.L. Machine Learning in Geosciences and Remote Sensing. *Geosci. Front.* **2016**, *7*, 3–10. [[CrossRef](#)]
11. Rigol, J.P.; Jarvis, C.H.; Stuart, N. Artificial Neural Networks as a Tool for Spatial Interpolation. *Int. J. Geogr. Inf. Sci.* **2001**, *15*, 323–343. [[CrossRef](#)]
12. Snell, S.E.; Gopal, S.; Kaufmann, R.K. Spatial Interpolation of Surface Air Temperatures using Artificial Neural Networks: Evaluating their use for Downscaling GCMs. *J. Clim.* **2000**, *13*, 886–895. [[CrossRef](#)]
13. Xu, C.; Chen, H.; Wang, J.; Guo, Y.; Yuan, Y. Improving Prediction Performance for Indoor Temperature in Public Buildings based on a Novel Deep Learning Method. *Build. Environ.* **2019**, *148*, 128–135. [[CrossRef](#)]
14. Xue, H.; Jiang, W.; Miao, C.; Yuan, Y.; Ma, F.; Ma, X.; Wang, Y.; Yao, S.; Xu, W.; Zhang, A.; et al. DeepFusion: A Deep Learning Framework for the Fusion of Heterogeneous Sensory Data. In Proceedings of the 20th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), Catania, Italy, 2–5 July 2019; pp. 151–160.
15. Ma, J.; Ding, Y.; Cheng, J.C.; Jiang, F.; Wan, Z. A Temporal-spatial Interpolation and Extrapolation Method based on Geographic Long Short-Term Memory Neural Network for PM_{2.5}. *J. Clean. Prod.* **2019**, *237*, 117729. [[CrossRef](#)]
16. Ma, R.; Liu, N.; Xu, X.; Wang, Y.; Noh, H.Y.; Zhang, P.; Zhang, L. Fine-Grained Air Pollution Inference with Mobile Sensing Systems: A Weather-Related Deep Autoencoder Model. *Proc. ACM Interact. Mobile Wearable Ubiquitous Technol. (IMWUT)* **2020**, *4*, 1–21. [[CrossRef](#)]
17. Najjar, N.; Gupta, S.; Hare, J.; Kandil, S.; Walthall, R. Optimal Sensor Selection and Fusion for Heat Exchanger Fouling Diagnosis in Aerospace Systems. *IEEE Sens. J.* **2016**, *16*, 4866–4881. [[CrossRef](#)]
18. Palmer, K.A.; Bollas, G.M. Sensor Selection Embedded in Active Fault Diagnosis Algorithms. In *IEEE Transactions on Control Systems Technology*; IEEE: Piscataway, NJ, USA, 2019.
19. Jin, H.; Su, L.; Chen, D.; Nahrstedt, K.; Xu, J. Quality of Information Aware Incentive Mechanisms for Mobile Crowd Sensing Systems. In Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), Hangzhou, China, 22–25 June 2015; pp. 167–176.
20. Jawaid, S.T.; Smith, S.L. Submodularity and Greedy Algorithms in Sensor Scheduling for Linear Dynamical Systems. *Automatica* **2015**, *61*, 282–288. [[CrossRef](#)]
21. Vitus, M.P.; Zhang, W.; Abate, A.; Hu, J.; Tomlin, C.J. On Efficient Sensor Scheduling for Linear Dynamical Systems. *Automatica* **2012**, *48*, 2482–2493. [[CrossRef](#)]
22. Gupta, V.; Chung, T.H.; Hassibi, B.; Murray, R.M. On a Stochastic Sensor Selection Algorithm with Applications in Sensor Scheduling and Sensor Coverage. *Automatica* **2006**, *42*, 251–260. [[CrossRef](#)]
23. Wu, J.; Jia, Q.S.; Johansson, K.H.; Shi, L. Event-based Sensor Data Scheduling: Trade-off between Communication Rate and Estimation Quality. *IEEE Trans. Autom. Control* **2012**, *58*, 1041–1046. [[CrossRef](#)]
24. Mo, Y.; Ambrosino, R.; Sinopoli, B. Sensor Selection Strategies for State Estimation in Energy Constrained Wireless Sensor Networks. *Automatica* **2011**, *47*, 1330–1338. [[CrossRef](#)]
25. Hare, J.Z.; Gupta, S.; Wettergren, T.A. POSE: Prediction-based Opportunistic Sensing for Energy Efficiency in Sensor Networks using Distributed Supervisors. *IEEE Trans. Cybern.* **2017**, *48*, 2114–2127. [[CrossRef](#)]
26. Lundberg, S.M.; Lee, S.I. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems 30*; Curran Associates Inc.: Red Hook, NY, USA, 2017; pp. 4765–4774.
27. Chen, T.; Guestrin, C. XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), San Francisco, CA, USA, 13–17 August 2016; pp. 785–794.
28. Keogh, E.; Lonardi, S.; Ratanamahatana, C.A.; Wei, L.; Lee, S.H.; Handley, J. Compression-based Data Mining of Sequential Data. *Data Min. Knowl. Discov.* **2007**, *14*, 99–129. [[CrossRef](#)]
29. Lin, J.; Keogh, E.; Wei, L.; Lonardi, S. Experiencing SAX: A Novel Symbolic Representation of Time Series. *Data Min. Knowl. Discov.* **2007**, *15*, 107–144. [[CrossRef](#)]
30. Oswal, S.; Singh, A.; Kumari, K. Deflate Compression Algorithm. *Int. J. Eng. Res. Gen. Sci.* **2016**, *4*, 430–436.
31. Fortin, F.A.; De Rainville, F.M.; Gardner, M.A.G.; Parizeau, M.; Gagné, C. DEAP: Evolutionary Algorithms Made Easy. *J. Mach. Learn. Res.* **2012**, *13*, 2171–2175.

32. Luke, S.; Panait, L. Fighting Bloat with Nonparametric Parsimony Pressure. In Proceedings of the 7th International Conference on Parallel Problem Solving from Nature (PPSN), Granada, Spain, 7–11 September 2002; pp. 411–421.
33. Borda, J.D. Mémoire sur les élections au scrutin. In *Histoire de l'Académie Royale des Sciences pour 1781*; Gallica: Paris, France, 1784
34. Satopaa, V.; Albrecht, J.; Irwin, D.; Raghavan, B. Finding a “kneedle” in a haystack: Detecting knee points in system behavior. In Proceedings of the 31st International Conference on Distributed Computing Systems (ICDCS) Workshops, Minneapolis, MN, USA, 20–24 June 2011; pp. 166–171.
35. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
36. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. Pytorch: An Imperative Style, High-performance Deep Learning Library. *Adv. Neural Inform. Proces. Syst.* **2019**, *32*, 8026–8037.
37. Friedman, J.H. Greedy Function Approximation: A Gradient Boosting Machine. *Ann. Stat.* **2001**, *29*, 1189–1232. [[CrossRef](#)]
38. Bergstra, J.; Yamins, D.; Cox, D. Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. In Proceedings of the 30th International Conference on Machine Learning (ICML), Atlanta, GA, USA, 16–21 June 2013; pp. 115–123.
39. Koenker, R.; Hallock, K.F. Quantile Regression. *J. Econ. Perspect.* **2001**, *15*, 143–156. [[CrossRef](#)]